SPACE COMPLEXITY

TUESDAY OCTOBER 7

MEASURING SPACE COMPLEXITY

Measure space complexity by marking the right-most tape cell that is reached on the computation

Definition: Let M be a deterministic TM that halts on all inputs. The space complexity of M is the function f : N → N, where f(n) is the rightmost tape position that M reaches on any input of length n.

Definition: Let M be a non-deterministic TM that halts on all inputs. The space complexity of M is the function f : N → N, where f(n) is the rightmost tape position that M reaches on any branch if its computation on any input of length n.

Definition: SPACE(t(n)) = { L | L is a language decided by a O(t(n)) space deterministic Turing Machine }

Definition: NSPACE(t(n)) = { L | L is a language decided by a O(t(n)) space non-deterministic Turing Machine }

3SAT ∈ **SPACE(n)**

Assume a branch of a non-looping non-deterministic computation uses space k

How many time steps can this branch have?

$$k\,|Q|\,|\Gamma|^k = 2^{O(k)}$$

SAVITCH'S THEOREM

Theorem: For a function f where f(n) ≥ n

$$NSPACE(f(n)) \subseteq SPACE(f(n)^2)$$

Proof:

Let N be a non-deterministic TM with space complexity f(n)

Construct a deterministic machine M that simply tries each branch of N

Since each branch of N uses space at most f(n), then M uses space at most f(n)

We need to simulate a non-deterministic computation while saving as much space as possible

Idea: Given two configurations c_1 and c_2 together with a number t, test whether N can get from c_1 to c_2 in t steps and space f(n)

BOOL CANYIELD(c_1 and c_2, integer t, space f(n))

If t = 1, then test directly whether $c_1 = c_2$ or whether c_1 yields c_2 in one step. Accept if yes, Reject if no.

If t > 1, then for each configuration c_m of size f(n)
Accept if
CANYIELD(c_1,c_m,t/2) && CANYIELD(c_m,c_2,t/2)

Reject

SAVITCH'S THEOREM

Theorem: For a function f where f(n) ≥ n

$$NSPACE(f(n)) \subseteq SPACE(f(n)^2)$$

Proof:

Let N be a non-deterministic TM with space complexity f(n)

We modify N so that when it accepts, it clears its tape and moves the head to the leftmost cell

Construct a deterministic machine M that on input w, simply returns CANYIELD(c_0,c_{accept}, $2^{df(n)}$, f(n))

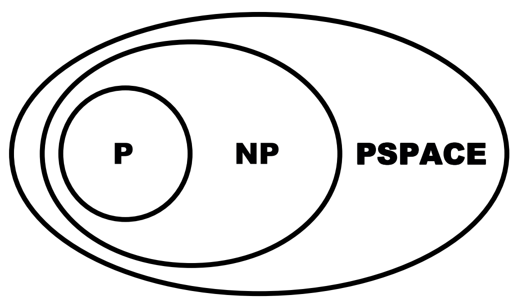
$$\text{PSPACE} = \bigcup_{k \in N} \text{SPACE}(n^k)$$

$$\text{NPSPACE} = \bigcup_{k \in N} \text{NSPACE}(n^k)$$

PSPACE = NPSPACE

P $\subseteq$ PSPACE?

NP $\subseteq$ PSPACE?



Assume a deterministic Turing machine that halts on all inputs runs in space f(n)

Question: At most how many time steps can this machine run for?

$$f(n)\ |Q|\ |\Gamma|^{f(n)} = 2^{O(f(n))}$$

$$\text{EXPTIME} = \bigcup_{k \in \mathbb{N}} \text{TIME}(2^{n^k})$$

PSPACE ⊆ EXPTIME

P ⊆ NP ⊆ PSPACE ⊆ EXPTIME

P ≠ EXPTIME

Definition: Language B is PSPACE-complete if:

1. B ∈ PSPACE
2. Every A in PSPACE is poly-time reducible to B (i.e. B is PSPACE-hard)

QUANTIFIED BOOLEAN FORMULAS

$\exists x \exists y\ [\ x \vee \neg y\ ]$

$\forall x\ [\ x \vee \neg x\ ]$

$\forall x\ [\ x\ ]$

$\forall x \exists y\ [\ (x \vee y) \wedge (\neg x \vee \neg y)\ ]$

Definition: A fully quantified Boolean formula (or a sentence) is a Boolean formula in which every variable is quantified

$$\exists x \exists y \, [\, x \vee \neg y \,]$$

$$\forall x \, [\, x \vee \neg x \,]$$

$$\forall x \, [\, x \,]$$

$$\forall x \exists y \, [\, (x \vee y) \wedge (\neg x \vee \neg y) \,]$$

TQBF = { ϕ | ϕ is a true fully quantified Boolean formula}

Theorem: TQBF is PSPACE-complete

TQBF $\in$ PSPACE

T(ϕ):

1. If ϕ contains no quantifiers, then it is an expression with only constants, so simply evaluate ϕ

2. If $\phi = \exists x\, \psi$, recursively call T on ψ, first with x = 0 and then with x = 1. Accept if either one of them is true.

3. If $\phi = \forall x\, \psi$, recursively call T on ψ, first with x = 0 and then with x = 1. Accept if both of them are true.

WWW.FLAC.WS

Read Chapters 8.1 and 8.2 of the book for next time